SANDIA REPORT

SAND2012-0491
Unlimited Release
Printed August 2012

Surrogate models for mixed discrete-continuous variables

Laura P. Swiler, Patricia D. Hough, Peter Qian, Xu Xu, Curtis Storlie, Herbert Lee

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

SAND2012-0491
Unlimited Release
Printed August 2012

Surrogate models for mixed discrete-continuous variables

Laura P. Swiler
Optimization and Uncertainty Quant.
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1318
lpswile@sandia.gov

Patricia D. Hough
Quantitative Modeling and Analysis
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94451-9159
pdhough@sandia.gov

Peter Qian
Department of Statistics
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706
peterq@stat.wisc.edu

Xu Xu
Department of Statistics
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706
xuxu@stat.wisc.edu

Curtis Storlie
Statistics Department
Los Alamos National Laboratory
P.O. Box 1663, MS F600
Los Alamos, NM 87545
storlie@lanl.gov

Herbert Lee
Applied Mathematics and Statistics
University of California, Santa Cruz
Basked School of Engineering
1156 High St, MS SOE2
Santa Cruz, CA 95064
herbie@soe.ucsc.edu

Abstract

Large-scale computational models have become common tools for analyzing complex man-made systems. However, when coupled with optimization or uncertainty quantification methods in order to conduct extensive model exploration and analysis, the computational expense quickly becomes intractable. Furthermore, these models may have both continuous and discrete parameters. One common approach to mitigating the computational expense is the use of response surface approximations. While well developed for models with continuous parameters, they are still new and largely untested for models with both continuous and discrete parameters. In this work, we describe and investigate the performance of three types of response surfaces developed for mixed-variable models: Adaptive Component Selection and Shrinkage Operator, Treed Gaussian Process, and Gaussian Process with Special Correlation Functions. We focus our efforts on test problems with a small number of parameters of interest, a characteristic of many physics-based engineering models. We present the results of our studies and offer some insights regarding the performance of each response surface approximation method.

Acknowledgments

We thank the Laboratory Directed Research and Development (LDRD) team for many fruitful discussions on this topic and the ongoing support of this project. The LDRD team and associated colleagues includes: William Hart, John Siirola, Jean-Paul Watson, Genetha Gray, Cynthia Phillips, Ali Pinar, and David Woodruff (UC Davis). We also thank management for support of this project, specifically M. Daniel Rintoul and the LDRD office at Sandia National Laboratories. Finally, we would like to thank Ken Perano for writing the C++ software that encodes the test functions we used to evaluate the surrogate approaches.

Contents

Appendix

List of Figures

Chapter 1

Introduction

Heterogeneous system of systems (HSoS) models have emerged as a promising approach for describing large, complex systems with distinct, independent subsystems. However, the current methods for analyzing such large-scale models with data and scenario uncertainties are computationally expensive. The overall goal of LDRD Project 09-0372, "Optimization of Large-Scale Heterogeneous System-of-Systems Models", was to develop approaches and tools for predicting and planning future behavior of complex man-made systems using these large-scale system models. More specifically, the focus was to develop optimization methods that exploit mathematical structure inherent in the HSoS models in order to efficiently analyze them.

One key to successfully realizing this goal involves the construction of surrogates for the computationally expensive models. A surrogate can take many forms, but in this context we mean a meta-model or response surface approximation built from a limited amount of data generated by the computationally expensive model. The purpose of the surrogate model is to increase the efficiency of analyses that require frequent model interrogations such as optimization and uncertainty quantification, thereby improving the tractability of these simulation-based analyses. The particular challenge we address in this work is developing surrogates for models that have both continuous and discrete variables.

There are two classes of models that have discrete and continuous variables that are of interest to us:

Engineering Models The engineering models typically involve a small number of variables of interest (e.g. tens of variables), but are characterized by computationally expensive equation solvers, such as partial differential equation solvers to model heat transfer, shock physics, etc. Many of the variables are continuous, but there can be discrete variables that represent modeling choices (alternative plausible models) and design choices (e.g. discrete choices of materials, components, or operational settings).

Large scale HSoS models These models simulate the behavior of complex systems such as the electric power grid, the transportation system, and logistics systems. These models are typically composed of many constitutive system models, and may have very large numbers of discrete and continuous variables. Discrete variables may represent quantities such as inventory levels, yes/no decisions, number of units transferred from one point in a network to another, etc.

The main focus of our work has been on the first class of models, Engineering Models. Scalability remains a key challenge for developing mixed variable surrogates for large-scale HSoS models.

The major challenge in using surrogates for mixed variable problems is the discrete variables. Typically, in surrogate models constructed over continuous variables, there is the assumption of continuity: as a continuous variable varies by a small amount, the response is assumed to vary smoothly. This is not always the case, and there are surrogate methods that can handle discontinuities in responses, but most surrogates (e.g. polynomial regression, splines, Gaussian process models, etc.) rely on assumptions of continuity.

With discrete variables, we do not necessarily have continuous behavior. For example, if one is varying a logistics system having two service centers to three service centers, the behavior of the system in terms of response time, average numbers of customers in a queue, etc. may be quite different. Similarly, if a discrete variable representing a design choice varies from choice A to choice B, the system may respond in a fundamentally different manner. In the worst case scenario, we would need to construct a separate surrogate model for each combination of the discrete variable settings. This can be very computationally expensive itself.

In this work, we consider three approaches for constructing mixed variable surrogates. They have their roots in response surface modelling for continuous problems and tractably incorporate discrete variables based in a manner that relies on some simplifications and additional assumptions.

1.1 Related Work

The analysis of many physical and engineering problems involves running complex computational models. With problems of this type, it is important to understand the relationships between the input variables, whose values are often imprecisely known, and the output. However, a computational model that sufficiently represents reality is often very costly to run. Thus, there has been a strong interest to develop "emulators" or "metamodels" which are surrogate models of the simulation (e.g. a statistical model of the simulator output).

When the models are computationally demanding, meta-model approaches to their analysis have been shown to be very useful. For example, one standard approach in the literature is to develop an emulator that is a stationary smooth Gaussian process[20, 10, 21]. There are many good overview articles which compare various metamodel strategies. For example, Storlie et al. compare various smoothing predictors and nonparametric regression approaches in [26, 27]. Simpson et al. provide an excellent overview not just of various statistical metamodel methods but also approaches which use low-fidelity models as surrogates for high fidelity models [23]. This paper also addresses the use of surrogates in design optimization, which is a popular research area for computationally expensive disciplines such as computational fluid dynamics in aeronautical engineering design. Haftka and his students developed an approach which uses "ensembles" of emulators or hybrid emulators [31, 30]. The advantage of these types of hybrid or ensembles of emulators is

that better performance can be obtained. For example, one can select the best surrogate for various features or responses, or one can use weighted model averaging of surrogates.

Generally, the metamodels utilized in surrogate modeling do not explicitly allow for categorical input variables. Hence, they must be handled in one of two ways. One option is to order these categorical inputs in some way and treat them as continuous variables when creating a metamodel. In some cases, this can lead to undesireable and misleading results. The other option is categorical regression. In this approach, a separate surrogate model is constructed over the continuous variables for each possible combination of the discrete variable values. This approach has the advantage that the surrogate is only constructed on the continuous variables, conditional on a particular combination of discrete values. This approach may work fine if there are only a few discrete variable with a few values, but will quickly become infeasible as one increases the number of discrete variables and/or the number of levels per variable [15]. It is clear that a more appropriate and efficient treatment of categorical inputs is needed.

The remaining sections of this report are as follows. Section 2 outlines three approaches for constructing mixed variable surrogates. Section 3 describes a testbed that we developed for the purposes of testing the approaches. Section 4 provides results of the surrogates on several test problems, and Section 5 summarizes the outcome.

Chapter 2

Mixed Surrogate Approaches

This chapter describes four classes of methods that we investigated to generate surrogate models for mixed discrete-continuous variable problems. These four classes of methods are:

ACOSSO: ACOSSO, the Adaptive COmponent Selection and Smoothing Operator, is a specialized smoothing spline model[25]. It uses the smoothing spline ANOVA decomposition to separate the underlying function into simpler functional components (i.e., main effects, two-way interactions, etc.) then explicitly estimates these functional components in one optimization. The estimation proceeds by optimizing the likelihood subject to a penalty on each of the functional components. Each component involving continuous predictors has a penalty on its roughness and overall trend, each component involving discrete predictors has a penalty on its magnitude (L2 norm), while interaction components involving both discrete and continuous predictors receive a combination of these penalties.

Gaussian Processes with special correlation functions: Gaussian process models are powerful emulators for computer models. A Gaussian process model is defined by its mean and covariance function. The covariance function specifies how the response between two points is related: the idea is that points close together in input space will tend to have responses that are similar. Typically, the covariance function is a function of the distance between the points. Qian et al. have studied a variety of covariance functions that represent the covariance between discrete points [17][34]. They provide several correlation functions that are appropriate to use for mixed variable problems: we investigated the exchangeable correlation (EC), the multiplicative correlation (MC), and the unrestricted correlation function (UC). For comparison, we also looked at the Individual Kriging (IK) model which involves constructing a separate Gaussian process surrogate over the continuous variables for each combination of discrete variables. This is similar to categorical regression. Finally, we looked at both Gaussian correlation functions which are most typically used in Gaussian process models and the Wendland correlation function, which has compact support.

TGP: TGP, the treed Gaussian Process model, is an approach which allows different Gaussian process models (GPs) to be constructed on different partitions of the space [5][6]. This approach naturally lends itself to discrete variables, where the partitioning can be done between different values or sets of discrete variables. In TGP, the discrete or categorical variables are converted to a series of binary variables. The binary variables are then what are partitioned upon: they become the "nodes" of the tree [7].

2.1 Adaptive COmponent Selection and Shrinkage Operator (ACOSSO)

The Adaptive COmponent Selection and Shrinkage Operator (ACOSSO) estimate [25] was developed under the smoothing spline ANOVA (SS-ANOVA) modeling framework. As it is a smoothing type method, ACOSSO works best when the underlying function is somewhat smooth. The type of splines we are using involve the minimization of an objective function involving a sum-of-squares error term, similar to regression modeling. However, in the objective function for the splines, there are additional terms which can be viewed as regularization terms: these penalty terms help smooth the function and they also help perform variable selection. In the ACOSSO implementation, there is a penalty on functions of the categorical predictors. This penalty formulation provides a variable selection and automatic model reduction: it encourages some of the terms in the objective function to be zero, removing certain discrete variables or levels of discrete variables from the formulation. To facilitate the description of ACOSSO, we first review the univariate smoothing spline. We then describe the extension to multiple inputs while assuming that the predictors are continuous. Lastly, we introduce the treatment of categorical predictors and the ACOSSO estimator.

Univariate Smoothing Splines. Let x_n, $n = 1,\ldots,N$, denote the n^{th} observation of a univariate predictor x and let $y_n = f(x_n) + \varepsilon_n$ denote the observed output from a model f, where ε_n is an error term which may account for errors (usually small) incurred by the numerical method used to solve for f. Without loss of generality, we restrict attention to the domain $[0,1]$. We can always rescale the input x to this domain via a transformation. Assume that the unknown function f to be estimated belongs to 2^{nd} order Sobolev space $\mathscr{S}^2 = \{f : f, f' \text{ are absolutely continuous and } f'' \in \mathscr{L}^2[0,1]\}$. The smoothing spline estimate is given by the element $f \in \mathscr{S}^2$ that minimizes

$$\frac{1}{n}\sum_{n=1}^{N}[y_n - f(x_n)]^2 + \lambda \int_0^1 \left[f''(x)\right]^2 dx. \tag{2.1.1}$$

The penalty term on the right of (2.1.1) is an overall measure of the magnitude of the curvature (roughness) of the function over the domain. Thus, the tuning parameter λ controls the trade-off in the resulting estimate between smoothness and fidelity to the data; large values of λ will result in smoother functions while smaller values of λ result in rougher functions that more closely match the data. Generally, λ is chosen by generalized cross validation (GCV) [3], m-fold CV [11], or related methods ([4], pp. 239-243 and [9], pp. 42-52). The minimizer of Eq. (2.1.1) is technically called the cubic smoothing spline because the solution can be shown to be a natural cubic spline with knots at all of the distinct values of x_n, $n = 1,\ldots,N$ ([4], p. 230).

Multivariate Smoothing Splines. Now consider a vector of predictors $\mathbf{x} = [x_1,\ldots,x_I]'$. The simplest extension of smoothing splines to multiple inputs is the additive model [9]. For instance, assume that

$$f \in \mathscr{F}_{add} = \{f : f(\mathbf{x}) = \sum_{i=1}^{I} g_i(x_i), g_i \in \mathscr{S}^2\}, \tag{2.1.2}$$

i.e., $f(\mathbf{x}) = \sum_{i=1}^{I} g_i(x_i)$ is a sum of univariate functions. Let $\mathbf{x}_n = [x_{n,1},\ldots,x_{n,I}]'$ be the n^{th} observation of a multivariate predictor $\mathbf{x}$, $n = 1,\ldots,N$, and $y_n = f(\mathbf{x}_n) + \varepsilon_n$. The additive smoothing spline

estimate of f is the minimizer of

$$\frac{1}{n}\sum_{n=1}^{N}[y_n - f(\mathbf{x}_n)]^2 + \sum_{i=1}^{I}\lambda_i \int_0^1 \left[g_i''(x_i)\right]^2 dx_i \tag{2.1.3}$$

over $f \in \mathscr{F}_{add}$. The minimizer of the expression in Eq. (2.1.3), $\hat{f}(\mathbf{x}) = \sum_{i=1}^{I}\hat{g}_i(x_i)$, takes the form of a natural cubic spline for each of the functional components $\hat{g}_i$. Notice that there are I tuning parameters (λ_i) for the additive smoothing spline. These are generally determined via some form of cross-validation. A generalization to two-way and higher order interaction functions can also be achieved in a similar manner; see [25] for the full details of including interactions in the SS-ANOVA framework. The minimizer of the expression in Eq. (2.1.3) can be obtained in an efficient manner via matrix algebra using results from reproducing kernel Hilbert space (RKHS) theory; for details see [32] or [8].

Discrete Predictors. A large advantage to the SS-ANOVA framework is the ability to handle categorical predictors with relative ease. To facilitate the discussion, we generalize our notation to the following. Assume that $\mathbf{x} = [x_1, \ldots, x_I]'$ are continuous on $[0,1]$ as previously in this section, while $\mathbf{z} = [z_1, \ldots, z_J]'$ are unordered discrete variables, and let the collection of the two types of predictors be denoted $\mathbf{w} = [\mathbf{x}', \mathbf{z}']'$. For simplicity, assume $z_j \in \{1, 2, \ldots, b_j\}$ for $j = 1, \ldots, J$ where the ordering of the integers representing the groups for z_j is completely arbitrary. For notational convenience, let $\mathscr{G}_i = \mathscr{S}^2$ for $i = 1, \ldots, I$. Also let the class of $\mathscr{L}^2$ functions on the domain of z_j (i.e., $\{1, 2, \ldots, b_j\}$) be denoted as $\mathscr{H}_j$ for $j = 1, \ldots, J$.

For simplicity, we can once again consider the class of additive functions,

$$\mathscr{F}_{add} = \{f : f(\mathbf{w}) = \sum_{i=1}^{I} g_i(x_i) + \sum_{j=1}^{J} h_j(z_j),\ g_i \in \mathscr{G}_i,\ h_j \in \mathscr{H}_j\}. \tag{2.1.4}$$

Let $\mathbf{w}_n = [x_{n,1}, \ldots, x_{n,I}, z_{n,1}, \ldots, z_{n,J}]'$ be the n^{th} observation of a multivariate predictor $\mathbf{w}$. The traditional additive smoothing spline is then the minimizer of

$$\frac{1}{N}\sum_{n=1}^{N}[y_n - f(\mathbf{w}_n)]^2 + \sum_{i=1}^{I}\lambda_i \int_0^1 \left[g_i''(x_i)\right]^2 dx_i \tag{2.1.5}$$

over $f \in \mathscr{F}_{add}$. Notice that in the traditional smoothing spline in (2.1.5) there is no penalty term on the functions of the categorical predictors (h_j).

Generalizing to the ACOSSO estimate. The COmponent Selection and Shrinkage Operator (COSSO) [12] penalizes on the sum of the semi-norms instead of the squared semi-norms as in Eq. (2.1.5). A semi-norm is a norm which can assign zero to some nonzero elements of the space, or alternatively can usually be thought of as a norm on a subset of the full space. In this case, all functions with zero second derivative (i.e., linear functions) will have zero penalty (i.e., semi-norm equal to zero). For ease of presentation, we will continue to restrict attention to the additive model. However, all of the following discussion applies directly to the two-way (or higher) interaction model as well.

The additive COSSO estimate, $\hat{f}(\mathbf{w}) = \sum \hat{g}_i(x_i) + \sum \hat{h}_j(z_j)$, is given by the function $f \in \mathscr{F}_{add}$ that minimizes

$$\frac{1}{N}\sum_{n=1}^{N}[y_n - f(\mathbf{w}_n)]^2 + \lambda_1 \sum_{i=1}^{I}\left\{\left[\int_0^1 g_i'(x_i)dx_i\right]^2 + \int_0^1 \left[g_i''(x_i)\right]^2 dx_i\right\}^{1/2} + \lambda_2 \sum_{j=1}^{J}\left\{\sum_{z_j=1}^{b_j} h_j^2(z_j)\right\}^{1/2}. \tag{2.1.6}$$

There are four key differences in the penalty term in Eq. (2.1.6) relative to the additive smoothing spline of Eq. (2.1.5). First, there is an additional term $\left[\int_0^1 g_i'(x_i)dx_i\right]^2$ in the penalty for continuous predictor functional components, which can also be written $[g_i(1) - g_i(0)]^2$, that penalizes the magnitude of the overall trend of the functional components g_i that correspond to continuous predictors. Second, there is now a penalty on the $\mathscr{L}^2$ norm of the h_j that correspond to the categorical predictors. Third, in contrast to the squared semi-norm in the additive smoothing spline, each term in the sum in the penalty in Eq. (2.1.6) can be thought of as a semi-norm over functions $g_i \in \mathscr{G}_i$ or $h_j \in \mathscr{H}_j$, respectively, (only constant functions have zero penalty). This has a similar effect to the Least Absolute Selection and Shrinkage Operator (LASSO) [29] for linear models in that it encourages some of the terms in the sum to be exactly zero. These terms are semi-norms over the g_i (or h_j); when such zeros result, $\hat{g}_i$ (or $\hat{h}_j$) is set to a constant, thus removing x_i (or z_j) from the estimate and providing automatic "model" selection/reduction. Fourth, the COSSO penalty only has two tuning parameters (three if two-way interactions are included), which can be chosen via GCV or similar means. This differs from the original COSSO [12] (i.e., all continuous predictors) where there is only one tuning parameter. It can be demonstrated analytically in that case, that the COSSO penalty with one tuning parameter gives as much flexibility as the penalty on the corresponding *squared* norms with I tuning parameters [12]. A similar flexibility is also true with the two tuning parameters in the mixed discrete/continuous predictor estimator of Eq. (2.1.6).

Finally, ACOSSO is a weighted version of COSSO, where a rescaled semi-norm is used as the penalty for each of the functional components. Specifically, we select as our estimate the function $f \in \mathscr{F}_{add}$ that minimizes

$$\frac{1}{N}\sum_{n=1}^{N}[y_n - f(\mathbf{w}_n)]^2 + \lambda_1 \sum_{i=1}^{I} v_i \left\{\left[\int_0^1 g_i'(x_i)dx_i\right]^2 + \int_0^1 \left[g_i''(x_i)\right]^2 dx_i\right\}^{1/2} + \lambda_2 \sum_{j=1}^{J} w_j \left\{\sum_{z_j=1}^{b_j} h_j^2(z_j)\right\}^{1/2}, \tag{2.1.7}$$

where the v_i, w_j, $0 < v_i, w_j \leq \infty$, are weights that can depend on an initial estimate of f which we denote $\tilde{f}$. Our implementation of ACOSSO takes $\tilde{f}$ to be the COSSO estimate of Eq. (2.1.6), in which λ_1 and λ_2 are chosen by the GCV criterion. We then use

$$\begin{aligned} v_i &= \left[\int_0^1 \tilde{g}_i^2(x_i)dx_i\right]^{-1} \text{ for } i = 1,\ldots,I \\ w_j &= \left(\frac{1}{b_j}\sum_{z_j=1}^{b_j}\tilde{h}_j^2(z_j)\right)^{-1} \text{ for } j = 1,\ldots,J. \end{aligned} \tag{2.1.8}$$

This allows for more flexible estimation (less penalty) on the functional components that show

more signal in the initial estimate. As shown in [25], this approach results in better performance on many test cases and more favorable asymptotic properties than COSSO.

The minimizer of the expression in Eq. (2.1.7) is obtained using an iterative algorithm and a RKHS framework similar to that used to find the minimizer of Eq. (2.1.5) in [32] and [8]. The optimization problem for the two-way interaction model can be posed in a similar way to Eq. (2.1.7); see [25] for more details on the interaction model and the computation of the solution. The two-way interaction model is used in the results of Chapter 4.

2.2 Gaussian Processes for Models with Quantitative and Qualitative Factors

This section describes a computationally efficient method developed in Zhou, Qian, and Zhou[34] for fitting Gaussian process models with quantitative and qualitative factors proposed in Qian, Wu, and Wu[17]. Consider a computer model with inputs $\mathbf{w} = (\mathbf{x}^t, \mathbf{z}^t)^t$, where $\mathbf{x} = (x_1, \ldots, x_I)^t$ consists of all the quantitative factors and $\mathbf{z} = (z_1, \ldots, z_J)^t$ consists of all the qualitative factors with z_j having b_j levels. The number of the qualitative levels of $\mathbf{z}$ is given by

$$m = \prod_{j=1}^{J} b_j. \tag{2.2.9}$$

Throughout, the factors in $\mathbf{z}$ are assumed to be qualitative but not ordinal. Gaussian process models with ordinal qualitative factors can be found in Section 4.4 of [17]. The response of the computer model at an input value $\mathbf{w}$ is modeled as

$$y(\mathbf{w}) = \mathbf{f}^t(\mathbf{w})\boldsymbol{\beta} + \varepsilon(\mathbf{w}), \tag{2.2.10}$$

where $\mathbf{f}(\mathbf{w}) = [f_1(\mathbf{w}), \ldots, f_p(\mathbf{w})]^t$ is a set of p user-specified regression functions, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^t$ is a vector of unknown coefficients and the residual $\varepsilon(\mathbf{w})$ is a stationary Gaussian process with mean 0 and variance σ^2. The model in (2.2.10) has a more general form than the standard Gaussian process model with only quantitative factors $\mathbf{x}$ given by

$$y(\mathbf{x}) = \mathbf{f}^t(\mathbf{x})\boldsymbol{\beta} + \varepsilon(\mathbf{x}), \tag{2.2.11}$$

where $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_p(\mathbf{x})]^t$ is a set of p user-specified regression functions depending on $\mathbf{x}$ only, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^t$ is a vector of unknown coefficients, and the residual $\varepsilon(\mathbf{x})$ is a stationary Gaussian process with mean 0, variance σ^2 and a correlation function for $\mathbf{x}$.

For m in (2.2.9), let $c_1, \ldots, c_m$ denote the m qualitative levels of $\mathbf{z}$ and let $\mathbf{w} = (\mathbf{x}^t, c_q)^t$ $(q = 1, \ldots, m)$ denote any input value. For two input values $\mathbf{w}_1 = (\mathbf{x}_1^t, c_1)^t$ and $\mathbf{w}_2 = (\mathbf{x}_2^t, c_2)^t$, the correlation between $y(\mathbf{w}_1)$ and $y(\mathbf{w}_2)$ is defined to be

$$\text{cor}\,[\varepsilon(\mathbf{w}_1), \varepsilon(\mathbf{w}_2)] = \tau_{c_1,c_2}\varphi(\mathbf{x}_1, \mathbf{x}_2), \tag{2.2.12}$$

where φ is the correlation function for the quantitative factors $\mathbf{x}$ in the model (2.2.10) and τ_{c_1,c_2} is the cross-correlation between the qualitative levels c_1 and c_2. The choice of φ is flexible. Two popular choices are the *Gaussian correlation function* [21]

$$\varphi(\mathbf{x}_1,\mathbf{x}_2)=\exp\left\{-\sum_{i=1}^{I}\phi_i(x_{1i}-x_{2i})^2\right\} \tag{2.2.13}$$

and the *spherical correlation function*

$$\varphi(\mathbf{x}_1,\mathbf{x}_2)=\prod_{i=1}^{I}(1-1.5\xi_i+0.5\xi_i^3), \tag{2.2.14}$$

where $\xi_i=\min\{1,\phi_i|x_{1i}-x_{2i}|\}$. To achieve numerical stability for the correlation matrix $\mathbf{R}$ in (2.2.10) with large size, one can also consider the Wendland's *compactly supported correlation function* [33]

$$\varphi(\mathbf{x}_1,\mathbf{x}_2)=(1-r)_+^{l+2}\left[(l^2+4l+3)r^2+(3l+6)r+3\right]/3, \tag{2.2.15}$$

where $r=\sqrt{\sum_{i=1}^{I}\phi_i(x_{1i}-x_{2i})^2}$ and $l=\lfloor I/2\rfloor+3$. The notation $(1-r)_+$ means that if $(r>1)$, then $(1-r)_+=0$, otherwise $(1-r)_+$=$(1-r)$. The unknown roughness parameters ϕ_i in (2.2.13), (2.2.14) or (2.2.15) will be collectively denoted as $\mathbf{\Phi}=\{\phi_i\}$. The $m\times m$ matrix $\mathbf{T}=\{\tau_{r,s}\}$, with entries being the cross-correlations among the qualitative levels, must be *positive definite with unit diagonal elements* in order for (2.2.12) to be a valid correlation function. This condition can be achieved in two ways. One way is to use the semi-definite programming techniques with positive definiteness constraints [17], which are computationally intensive. [34] provides a more efficient way for modeling $\mathbf{T}$ by using the hypersphere decomposition, originally introduced for modeling correlations in financial applications [18]. This method first applies a Cholesky-type decomposition to $\mathbf{T}$

$$\mathbf{T}=\mathbf{L}\mathbf{L}^t, \tag{2.2.16}$$

where $\mathbf{L}=\{l_{r,s}\}$ is a lower triangular matrix with strictly positive diagonal entries. Then, let $l_{1,1}=1$ and for $r=2,\ldots,m$, consider a *spherical coordinate system*

$$\begin{cases} l_{r,1}=\cos(\theta_{r,1}),\\ l_{r,s}=\sin(\theta_{r,1})\cdots\sin(\theta_{r,s-1})\cos(\theta_{r,s}),\ \text{for } s=2,\ldots,r-1,\\ l_{r,r}=\sin(\theta_{r,1})\cdots\sin(\theta_{r,r-2})\sin(\theta_{r,r-1}), \end{cases} \tag{2.2.17}$$

where $\theta_{r,s}\in(0,\pi)$. Denote by $\mathbf{\Theta}$ all $\theta_{r,s}$ involved in (2.2.17).

Suppose that the computer model under consideration is conducted at n different input values, $D_{\mathbf{w}}=(\mathbf{w}_1^0,\ldots,\mathbf{w}_n^0)$, with the corresponding response values denoted by $\mathbf{y}=(y_1,\ldots,y_n)^t$. The parameters in model (2.2.10) to be estimated are σ^2, $\boldsymbol{\beta}$, $\mathbf{\Phi}$ and $\mathbf{\Theta}$. The *maximum likelihood estimators* of these parameters are denoted by $\hat{\sigma}^2$, $\hat{\boldsymbol{\beta}}$, $\hat{\mathbf{\Phi}}$ and $\hat{\mathbf{\Theta}}$, respectively. The log-likelihood function of $\mathbf{y}$, up to an additive constant, is

$$-\frac{1}{2}\left[n\log(\sigma^2)+\log(|\mathbf{R}|)+(\mathbf{y}-\mathbf{F}\beta)^t\mathbf{R}^{-1}(\mathbf{y}-\mathbf{F}\beta)/\sigma^2\right], \tag{2.2.18}$$

where $\mathbf{F} = \left[\mathbf{f}(\mathbf{w}_1^0), \ldots, \mathbf{f}(\mathbf{w}_n^0)\right]^t$ is an $n \times p$ matrix and $\mathbf{R}$ is the correlation matrix with (i, j)th entry $\text{cor}\left[\varepsilon(\mathbf{w}_i^0), \varepsilon(\mathbf{w}_j^0)\right]$ defined in (2.2.12). Given $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$, $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}^2$ are

$$\hat{\boldsymbol{\beta}} = (\mathbf{F}^t \mathbf{R}^{-1} \mathbf{F})^{-1} \mathbf{F}^t \mathbf{R}^{-1} \mathbf{y}, \tag{2.2.19}$$

and

$$\hat{\sigma}^2 = (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})^t \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}})/n. \tag{2.2.20}$$

Plugging (2.2.19) and (2.2.20) into (2.2.18), $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Theta}}$ can be obtained as

$$(\hat{\boldsymbol{\Phi}}, \hat{\boldsymbol{\Theta}}) = \arg\min_{\boldsymbol{\Phi}, \boldsymbol{\Theta}} \{n \log(\hat{\sigma}^2) + \log(|\mathbf{R}|)\}. \tag{2.2.21}$$

The optimization problem in (2.2.21) only involves the constraints that $\theta_{r,s} \in (0, \pi)$ for $\hat{\boldsymbol{\Theta}}$ and $\phi_i \geq 0$ for $\hat{\boldsymbol{\Phi}}$. It can be solved by modifying the DACE toolbox in Matlab [13] to incorporate the reparameterization in (2.2.17). A small nugget term is added to the diagonals of $\mathbf{R}$ to mitigate potential singularity. The fitted model can be used to predict the response value y at any untried input value. Given $\hat{\sigma}^2$, $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{\Theta}}$, the *empirical best linear unbiased predictor* (EBLUP) of y at any input value $\mathbf{w}_0$ is

$$\hat{y}(\mathbf{w}_0) = \mathbf{f}^t(\mathbf{w}_0)\hat{\boldsymbol{\beta}} + \hat{\mathbf{r}}_0^t \hat{\mathbf{R}}^{-1} (\mathbf{y} - \mathbf{F}\hat{\boldsymbol{\beta}}), \tag{2.2.22}$$

where $\hat{\mathbf{r}}_0 = \left\{\text{cor}\left[\varepsilon(\mathbf{w}_0^0), \varepsilon(\mathbf{w}_1^0)\right], \ldots, \text{cor}\left[\varepsilon(\mathbf{w}_0^0), \varepsilon(\mathbf{w}_n^0)\right]\right\}^t$ and $\hat{\mathbf{R}}$ is the estimated correlation matrix of $\mathbf{y}$. Similarly to its counterpart for the Gaussian process model in (2.2.11) with quantitative factors, the EBLUP in (2.2.22) smoothly interpolates all the observed data points. Features of the function $\hat{y}(\mathbf{w})$ can be visualized by plotting the estimated functional main effects and interactions. Details of performing ANOVA decompositions can be found in [21].

In this work, we consider four methods for building Gaussian process models for a computer experiment with qualitative and quantitative factors.

- The individual Kriging method, denoted by *IK*. This method fits data associated with different qualitative levels separately using distinct Gaussian process models for the quantitative variables in (2.2.11).
- The *exchangeable correlation* method for the qualitative factors, denoted by *EC*. It assumes the cross-correlation $\tau_{r,s}$ in (2.2.12) to be

$$\tau_{r,s} = c \ (0 < c < 1) \quad \text{for } r \neq s.$$

- The *multiplicative correlation* method for the qualitative factors, denoted by *MC*. It assumes the cross-correlation $\tau_{r,s}$ in (2.2.12) to be

$$\tau_{r,s} = \exp\{-(\theta_r + \theta_s) I[r \neq s]\} \ (\theta_r, \theta_s > 0).$$

- The method proposed in (2.2.16) and (2.2.17) with an *unrestricted* correlation function for the qualitative factors, denoted by *UC*.

2.3 Treed Gaussian Processes (TGP)

In practice, many situations involving the emulation of computer models call for more flexibility than is reasonable under the common assumption of stationarity. However, a fully nonstationary model may be undesirable as well, because of the vastly increased difficulty of performing inference due to a nonstationary model's complexity. A good compromise can be local stationarity. A treed Gaussian process (TGP) (Gramacy and Lee, 2008) is designed to take advantage of local stationarity. It defines a treed partitioning process on the predictor space and fits distinct, but hierarchically related, stationary GPs to separate regions at the leaves. The treed form of the partition makes the model easily interpretable: having the treed partitions with separate GPs makes it easy to identify the GP model in each branch. At the same time, the partitioning results in smaller matrices for inversion than would be required under a standard GP model and thereby provides a nonstationary model that actually facilitates faster inference. Using a fully Bayesian approach allows for model averaging across the tree space, resulting in smooth and continuous fits when the data are not naturally partitioned. The partitions are fit simultaneously with the individual GP parameters using reversible jump Markov chain Monte Carlo, so that all parts of the model can be learned automatically from the data. The posterior predictive distribution thus takes into account uncertainty from the data, from the fitted parameters, and from the fitted partitions.

TGP inherits its partitioning scheme from simpler treed models such as CART (Breiman et al., 1984) and BCART (for Bayesian CART) (Chipman et al., 1998, 2002). Each uses recursive binary splits so that each branch of the tree in any of these models divides the predictor space in two, with multiple splits allowed on the same variable for full flexibility. Consider predictors $x \in R^P$ for some split dimension $p \in \{1,...,P\}$ and split value v, points with $x_p \leq v$ are assigned to the left branch, and points with $x_p > v$ are assigned to the right branch. Partitioning is recursive and may occur on any input dimension p, so arbitrary axis-aligned regions in the predictor space may be defined. Conditional on a treed partition, models are fit in each of the leaf regions. In CART the underlying models are constant in that only the mean and standard deviation of the real-valued outputs are inferred. TGP fits a Gaussian process Z_ν in each leaf ν using the following hierarchical model:

$$\begin{array}{lll} \mathbf{Z}_\nu|\beta_\nu,\sigma^2_\nu,\mathbf{K}_\nu \sim N_{n_\nu}(\mathbf{F}_\nu\beta_\nu,\sigma^2_\nu\mathbf{K}_\nu) & \beta_0 \sim N_m(\boldsymbol{\mu},\mathbf{B}) & \sigma^2_\nu \sim IG(\alpha_\sigma/2,q_\sigma/2) \\ \beta_\nu|\sigma^2_\nu,\tau^2_\nu,\boldsymbol{W},\beta_0 \sim N_m(\beta_0,\sigma^2_\nu\tau^2_\nu\mathbf{W}) & \mathbf{W}^{-1} \sim W((\rho\mathbf{V})^{-1},\rho) & \tau^2_\nu \sim IG(\alpha_\tau/2,q_\tau/2) \end{array} \quad (2.3.23)$$

where $\boldsymbol{F}_\nu = (\mathbf{1},\boldsymbol{X}_\nu)$ contains the data in that leaf. N, IG, and W are the Multivariate Normal, Inverse–Gamma, and Wishart distributions, respectively. $\boldsymbol{K}_\nu$ is the separable power family covariance matrix with a nugget.

Classical treed methods, such as CART, can cope quite naturally with categorical, binary, and ordinal inputs. For example, categorical inputs can be encoded in binary, and splits can be proposed with rules such as $x_p < 1$. Once a split is made on a binary input, no further process is needed, marginally, in that dimension. Ordinal inputs can also be coded in binary, and thus treated as categorical, or treated as real-valued and handled in a default way. This formulation presents an alternative to that of Section 2.2. While that formulation allows a powerful and flexible representation of qualitative inputs in the model, it does not allow for nonstationarity. TGP allows the

combination of qualitative inputs and nonstationary modeling.

Rather than manipulate the GP correlation to handle categorical inputs, the tree presents a more natural mechanism for such binary indicators. That is, they can be included as candidates for treed partitioning but ignored when it comes to fitting the models at the leaves of the tree. They must be excluded from the GP model at the leaves since, if ever a Boolean indicator is partitioned upon, the design matrix (for the GP or LM) would contain a column of zeros or ones and therefore would not be of full rank. The benefits of removing the Booleans from the GP model(s) go beyond producing full-rank design matrices at the leaves of the tree. Loosely speaking, removing the Boolean indicators from the GP part of the treed GP gives a more parsimonious model. The tree is able to capture all of the dependence in the response as a function of the indicator input, and the GP is the appropriate nonlinear model for accounting for the remaining relationship between the real-valued inputs and outputs. Further advantages to this approach include speed (a partitioned model gives smaller covariance matrices to invert) and improved mixing in the Markov chain when a separable covariance function is used since the size of the parameter space defining the correlation structure would remain manageable. Note that using a non-separable covariance function in the presence of indicators would result in a poor fit. Good range (d) settings for the indicators would not necessarily coincide with good range settings for the real-valued inputs. Finally, the treed model allows the practitioner to immediately ascertain whether the response is sensitive to a particular categorical input by tallying the proportion of time the Markov chain visited trees with splits on the corresponding binary indicator. A much more involved Monte Carlo technique (e.g., following Saltelli et al., 2008) would otherwise be required in the absence of the tree. If it is known that, conditional on having a treed process for the binary inputs (encoding categories), the relationship between the remaining real-valued inputs and the response is stationary, then we can improve mixing in the Markov chain further by ignoring the real valued inputs when proposing tree operations. Here we use the implementation developed by Broderick and Gramacy (2011).

Chapter 3

Testing and Assessment Approach

In order to evaluate and compare the four mixed-variable surrogate modeling approaches, we established a common experimental strategy that can be consistently applied to all of them. There are three primary components to which we paid particular attention. They are the test functions, the sample design used for surrogate construction, and the performance metrics. Each is described in the following subsections.

3.1 Test Functions

One gap we identified when collecting test functions on which to evaluate the four mixed-variable surrogate methods was the absence of a portable testbed that was easy to interface with all of the code implementations under consideration. In addition, we wanted a testbed that was generic enough to be re-used by ourselves and others to evaluate methods developed in the future. To these ends, we established a set of requirements for the testbed that include the following:

- fast-running evaluations in order to obtain results in a timely manner
- easy to compile, cross-platform compatibility for portability to a variety of computing platforms
- extendable in order to easily add new functions
- file-based input and output to provide a single easy interface to a variety of surrogate modeling, optimization, and uncertainty quantification software
- ability to control the number of discrete variables and the number of levels per discrete variable in order to test method scalability with respect to these features
- ability to control problem complexity in order to evaluate performance on a variety of problems

To meet our requirements, we developed a C++ testbed that can be made available to others who wish to perform similar testing. It can be compiled on any computing platform with standard C++ compilers. It reads a simple text input file that contains a list of parameter values and produces

a similar output file containing the corresponding function response. New functions can be added by writing a short C++ evaluate method for that function. The testbed in its current form consists of a set of defined functions and a polynomial generator that provide a range of fast-running functions with a variety of features. Both classes of functions are described further below.

3.1.1 Defined Functions

The first function we considered in our numerical experiments has one categorical variable with five levels. It also two continuous variables, both of which fall between the values of 0 and 1. This function has regions where the responses at the different categorical levels are very similar. This will allow us to evaluate how well the different surrogate approaches can resolve the different levels.

$$f(x) = \begin{cases} \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) & \text{if } x_1 = 1 \\ \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 12.0\sin(2\pi x_3 - \pi) & \text{if } x_1 = 2 \\ \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 0.5\sin(2\pi x_3 - \pi) & \text{if } x_1 = 3 \\ \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 8.0\sin(2\pi x_3 - \pi) & \text{if } x_1 = 4 \\ \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 3.5\sin(2\pi x_3 - \pi) & \text{if } x_1 = 5 \end{cases}$$

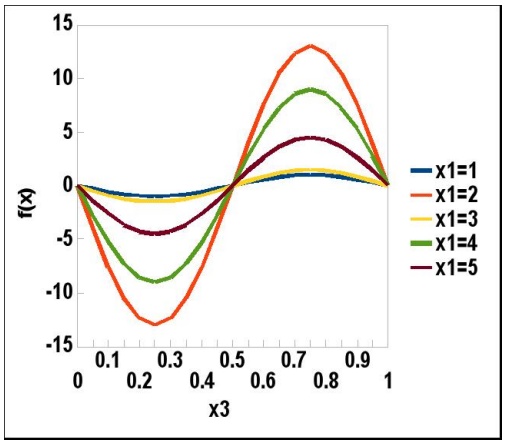

Figure 3.1. Test Function 2

The second function we considered is the Goldstein-Price function. It has one continuous variable and one discrete variable. The discrete variable, x_1, can take on the values of $-2, 0$, and 2. The continuous variable, x_2, ranges between the values of -2 and 2. It has notable parameter interactions, and the response spans multiple orders of magnitude.

$$f(x) = (1 + (x_1 + x_2 + 1)^2 * (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) * \\ (30 + (2x_1 - 3x_2)^2 * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$$

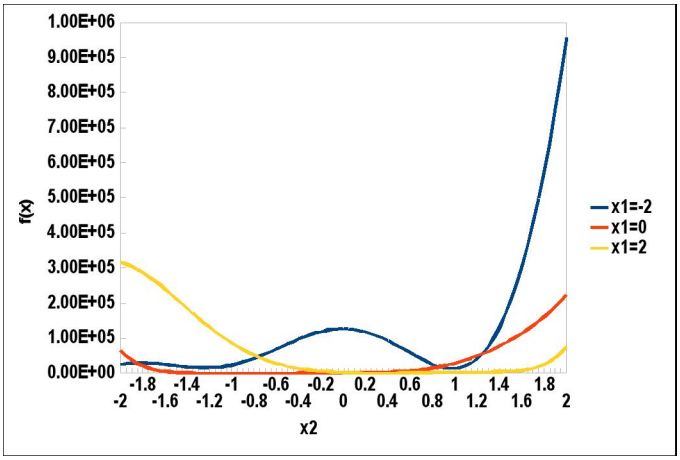

Figure 3.2. Goldstein Price Function

The testbed includes two other defined functions. We do not present results associated with these functions, however, so we defer their descriptions to Appendix A.

3.1.2 Polynomial Generator

The polynomial generator affords the greatest flexibility within the testbed. It can generate polynomials of degrees two through six with as many as fifteen variables. Any of those variables can be continous or categorical, and categorical variables can have an arbitrary number of levels. It is based on work by McDaniel and Ankenman [14] in which they develop a method for randomly generating polynomial functions. Their goal was to compare how different experimental design strategies worked as measured by how well a fitted surface based on the design models the true response. Their approach does not allow for direct control of the true response functions, but it does enable probabilistic control of characteristics such as response range and maximum number of stationary points.

Following McDaniel and Ankenmans's approach, the following characteristics can be controlled:

Effect sparcity refers to the fraction of factors being considered that appear in at least one term of the generated polynomial function. The user controls this by defining a range on the number of factors that could potentially appear in the definition of the polynomial surface. The algorithm randomly chooses some number of factors within that range.

Bumpiness refers to the prevalence of stationary points (maxima, minima, and inflection points) in the polynomial surface. This feature is not directly controlled by the user, but it is affected

by effect heredity, described below.

Response range describes the range of operability over which values of the response are defined, as established by lower and upper bounds on each factor. A range for the response is specified and is used to scale the polynomial response over the region of operability. This approach cannot guarantee that all response values are within the specified range, but it can ensure that most of the response range is within the specified targets.

Flatness is the extent to which local deviations in the polynomial surface are small with respect to the specified response range. It is specified by a scalar that controls the depth of local maxima and minima.

Effect heredity refers to the relationship between lower and higher order effects. The user establishes this by defining two sets of conditional probabilities. One set is the probability that a given term will appear provided its constituent factors appear in terms one order lower. The second set is the probability that a given term will appear provided its constituent factors do not appear in terms one order lower. This can be done for all main effects up to order six and for interaction terms up to order three.

Random error is included to represent the noise usually present in physical experiments, and the appropriate error distribution must be directly specified by the user. Since we are focused on (deterministic) computational experiments, we do not consider random error in this study.

For our numerical experiments, we use a 19-term fourth order polynomial. It has four parameters, two of which are continuous and two of which are discrete. The x_3 and x_4 are continuous variables that fall between 0 and 100, and x_1 and x_2 are discrete variables that have three levels, namely 20, 50, and 80. The polynomial is given by the following:

$$\begin{aligned} f(x) \;=\; & 53.3108 + 0.184901x_1 - 5.02914*10^{-6}x_1^3 + 7.72522*10^{-8}x_1^4 - \\ & 0.0870775x_2 - 0.106959x_3 + 7.98772*10^{-6}x_3^3 + 0.00242482x_4 + \\ & 1.32851*10^{-6}x_4^3 - 0.00146393x_1x_2 - 0.00301588x_1x_3 - \\ & 0.00272291x_1x_4 + 0.0017004x_2x_3 + 0.0038428x_2x_4 - 0.000198969x_3x_4 + \\ & 1.86025*10^{-5}x_1x_2x_3 - 1.88719*10^{-6}x_1x_2x_4 + 2.50923*10^{-5}x_1x_3x_4 - \\ & 5.62199*10^{-5}x_2x_3x_4 \end{aligned}$$

3.2 Sample Design

The accuracy of a response surface surrogate can be affected by the number of data points used to build it as well as how those points are chosen. Therefore, we vary the number and design of build points in our numerical experiments. All designs are based on Latin Hypercube designs (LHD) of the parameter space. We define n to be the number of LHD runs per qualitative level of the categorical variables and m to be the number of discrete levels (or combinations of levels). The

total number of points used to build each surrogate is mn. We consider $n = 10, 20, 40, 80$, and the sample design for each training set is constructed in three different ways.

Standard Latin Hypercube In this approach, one Latin Hypercube design of size mn is generated over all of the continuous parameters. It is then randomly split it into m groups of n runs, and each group is assigned a qualitative level of the categorical variables.

k Latin Hypercube In this approach, a separate Latin Hypercube design is generated for every given level of categorical variables. That is, we generate m independent Latin hypercube designs, each of size n and corresponding to one qualitative level.

Sliced Latin Hypercube This approach is based on recent work by Qian [16]. This design is a Latin Hypercube for the continuous factors and is sliced into groups of smaller Latin Hypercube designs associated with different categorical levels. In this case, we generate a sliced Latin hypercube design with m slices, where each slice of n runs corresponds to one qualitative level.

Because of the randomness associated with the LHS samples, we generate 10 replicate training sets for each combination of n and design type.

3.3 Comparison Metrics

Evaluating the performance of computational methods can be challenging, particularly with regard to the accuracy of the method. This is because the accuracy required for different applications of the method can vary. In this study, our primary focus is on gaining an understanding of the accuracy of mixed variable surrogates relative to each other, so we use a relatively fine-grained metric. In particular, we use mean squared error between surrogate predictions and true function values over a set of given points. For every replication of a given n and training design type, the mean squared errors (MSE) are calculated based on a testing set using a Latin hypercube design with 200 samples for each qualitative level. We then compare the mean and spread of the errors. Lower values of these quantities constitute better performance.

Chapter 4

Results

We present the results of our evaluation of the surrogate approaches presented in Chapter 2. Specifically, we compared the results of TGP, ACOSSO, and the Gaussian process model with the individual kriging, exchangeable correlation, multiplicative correlation, and unrestricted correlation functions. We do not present the results of categorical regression because the individual kriging approach is essentially that: it calculates a separate surrogate model for each categorical level. We applied these different methods to the test functions described in Chapter 3, comparing the results over different sample sizes of the sample designs presented in Section 3.2.

4.1 Test Function 2

We started by investigating Test Function 2 which has two continuous variables and one discrete variable with five levels. The results are shown in the following figures. These figures display boxplots based on 10 realizations of n samples per discrete level, where n is 10, 20, 40, or 80. The Y axis is the mean squared error (MSE) of the surrogate construction. The surrogates in all of these plots were constructed using sliced LHD designs. The Gaussian correlation function in (2.2.13) for the quantitative factors is used in the *IK*, *EC*, *MC* and *UC* methods. Figures 4.1-4.4 give the boxplots of the MSEs of the four methods for $n = 10, 20, 40, 80$. NOTE: the Y-axis scale is different on all of the four of theses Figures 4.1-4.4. Ideally, it would be nice to see the MSE plotted on the same scale so that it is easy to see the decrease in error as a function of the number of build samples. However, the MSE varied so dramatically for some of these results that keeping an MSE scale to allow for plotting maximum MSE values would result in the reader not seeing the differences in situations where the MSE was low. Thus, we have a different MSE scale on each plot.

Overall, ACOSSO does very well on this function and outperforms the other methods, especially at the smaller sample levels of $n = 10$ and $n = 20$. For the four GP correlation schemes, the *EC*, *MC* and *UC* methods outperform the *IK* method.

Generally, we found that sample design type (e.g. standard Latin Hypercube, kLHD, or sliced LHD) did not have a large effect on the MSE. Most of the results we will present in this report used the sliced LHD. However, there were some cases where the design did appear to make a difference. For example, TGP performed better for test function 2 with LHD, and the Gaussian process with

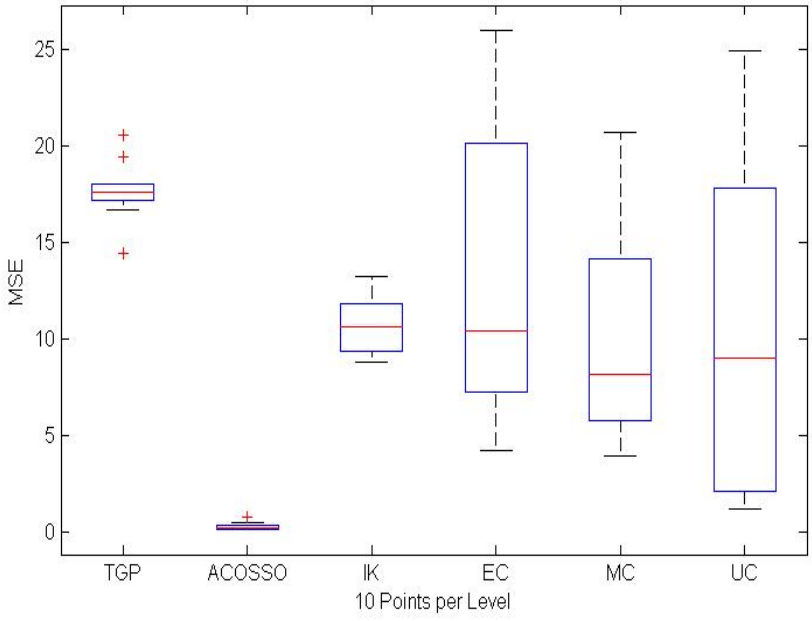

Figure 4.1. Test Function 2. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 10$ using the sliced LHD scheme.

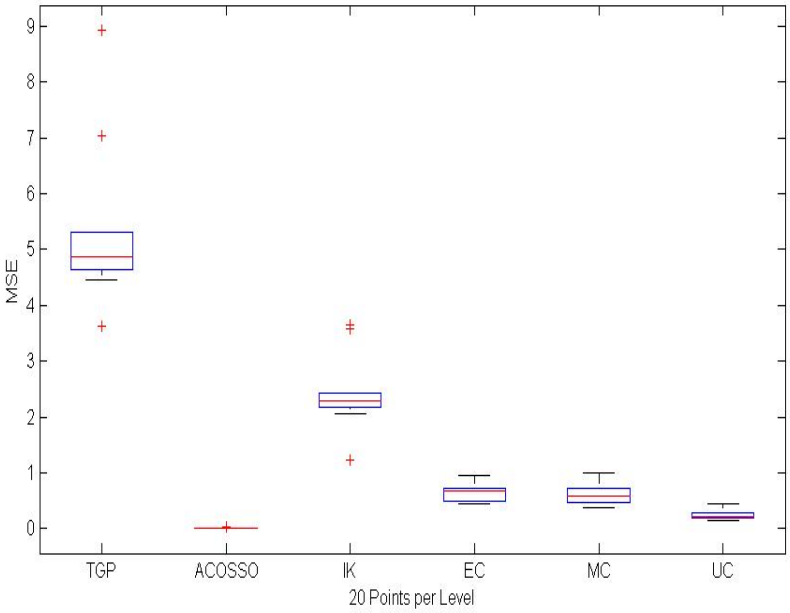

Figure 4.2. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 20$ using the sliced LHD scheme.

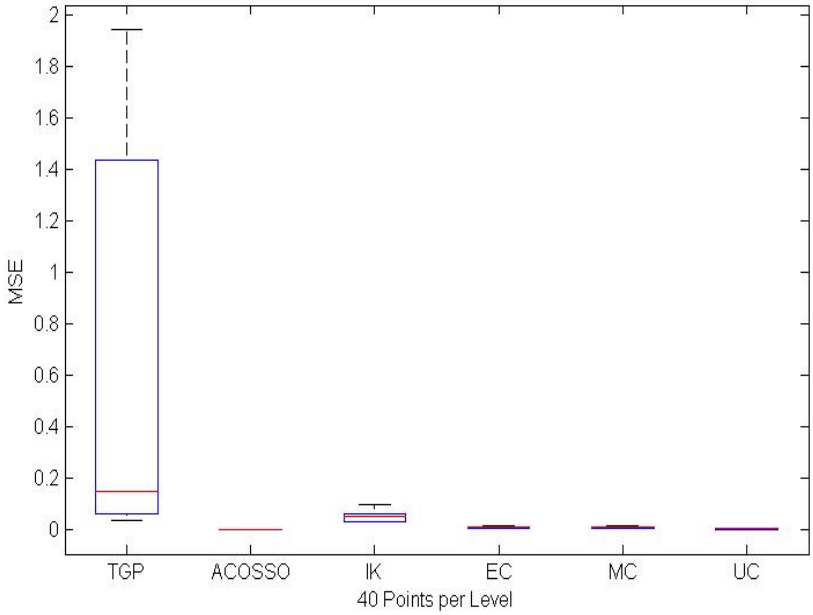

Figure 4.3. Test Function 2. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 40$ using the sliced LHD scheme.

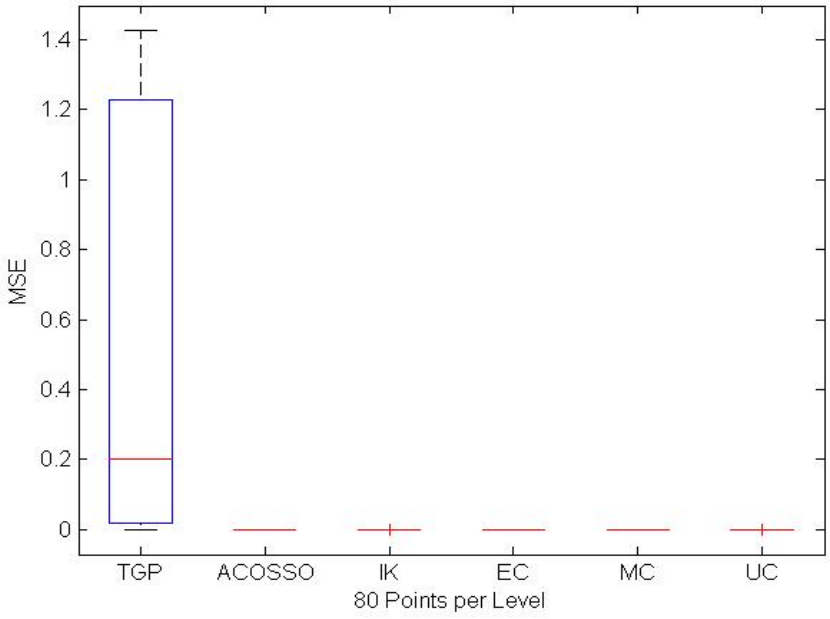

Figure 4.4. Test Function 2. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 80$ using the sliced LHD scheme.

the specialized correlation functions generally performed worse with LHD. Figure 4.5 shows the results of the LHD designs. Note that the MSE values for the *EC* correlation were so large, with a median MSE of 20,483, that they were omitted from this figure for scaling purposes.

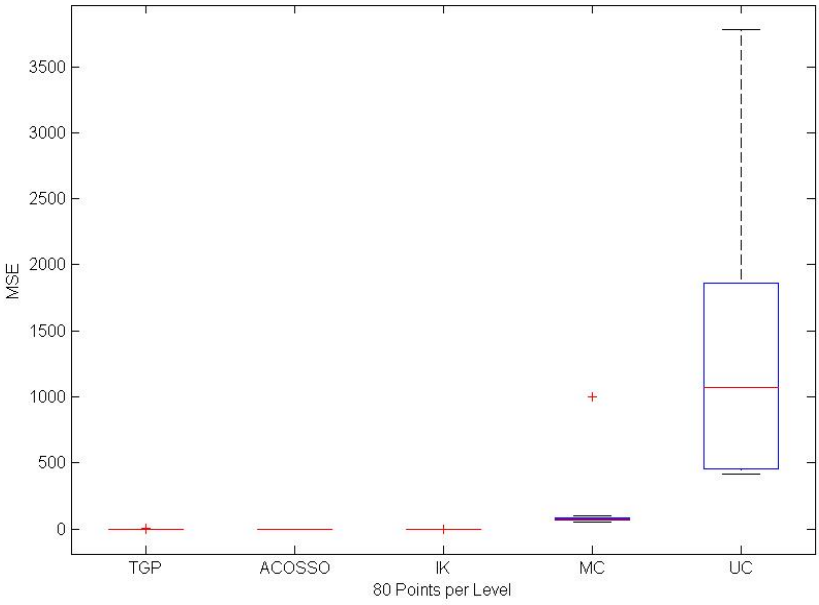

Figure 4.5. Test Function 2. Boxplots of the MSEs for the TGP, ACOSSO, IK, MC and UC methods with $n = 80$ using the standard LHD scheme.

Finally, we compared the Gaussian correlation function with the compactly supported Wendland correlation (2.2.15). Figure 4.6 shows a comparison of the MSE for the $n = 20$ case, using sliced lHD. For this particular test problem, the compact support does not appear to offer any advantage, and its performance is a little worse. In summary for Test Function 2: ACOSSO performed the best overall, the GP variations with *IK*, *EC*, *MC* and *UC* methods also performed well especially at $n = 40$ and $n = 80$, the *UC* method did not perform well with a standard LHD design, and the GP variations performed slightly better with a Gaussian vs. a Wendland correlation function.

4.2 Goldstein-Price

The Goldstein-Price results are shown in the following figures. Recall that the Goldstein-Price function has two variables, one of which we treated as a discrete variable and one of which we treated as continuous. This function varies by five orders of magnitude over the domain we chose, so we performed the surrogate construction in log space and the error is presented in log space.

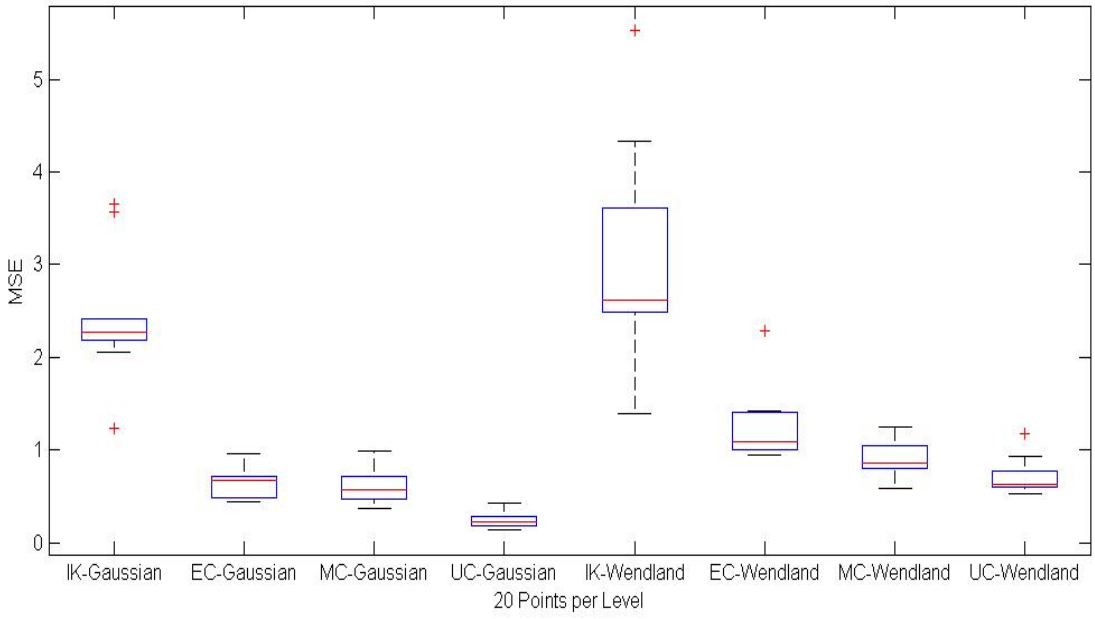

Figure 4.6. Test Function 2. Boxplots of the MSEs for the TGP, ACOSSO, IK, MC and UC methods with $n = 20$ using Gaussian vs. Wendland correlations.

Figures 4.7-4.10 give the boxplots of the MSEs of the four methods for $n = 10, 20, 40, 80$. In these figures, the Y axis is the mean squared error of the surrogate, but remember that the surrogate is constructed in log space so these are errors in log space. The surrogates in all of these plots were constructed using sliced LHD designs. For the Gaussian process model, the four variations of IK, EC, MC and UC methods all used the compact support Wendland correlation function in (2.2.15). For the Goldstein-Price function, the compactly supported correlation performed better than the Gaussian correlation function. For example, the average MSE for the UC method built on the $n = 80$ level was 9.22E-7 for the compact support correlation function while the average MSE for this same method and sample size was 1.05E-2 using the Gaussian correlation function. For this reason, we present the results using the compactly supported correlation function.

Overall, the variations of the Gaussian process model do very well on this function. ACOSSO also performs well, and the mean MSE from ACOSSO is close to the mean from the various GP methods. However, the variability of the ACOSSO results is slightly larger, as shown in Figures 4.7-4.10. Note that TGP has larger MSE at all sample levels. However, when we performed the surrogate construction in the original space without taking the logarithm of the Goldstein-Price function, TGP outperformed the other methods. This may be due to the ability of TGP to identify different regions of the space with different properties (e.g. the scale of the Goldstein-Price function is much smaller in the center of the domain than at the edges of the domain we are using for this case study).

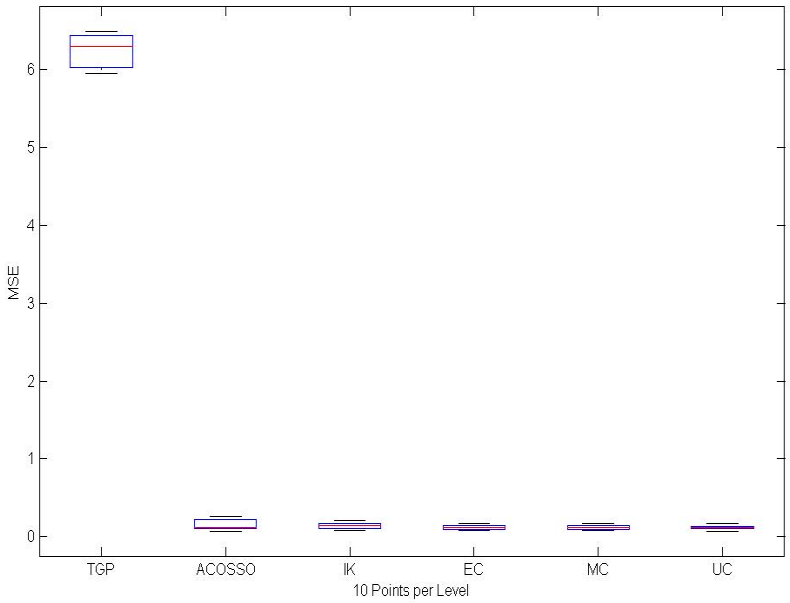

Figure 4.7. Goldstein-Price. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 10$ using the sliced LHD scheme.

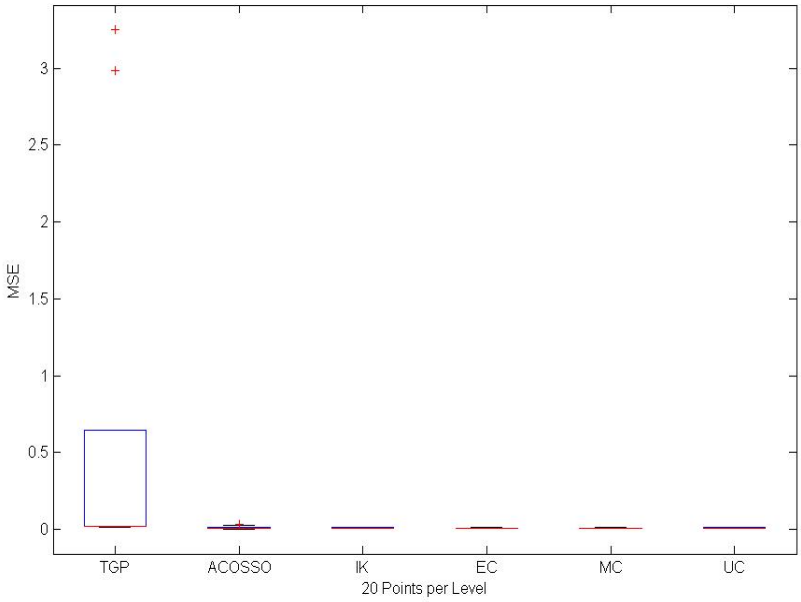

Figure 4.8. Goldstein-Price. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 20$ using the sliced LHD scheme.

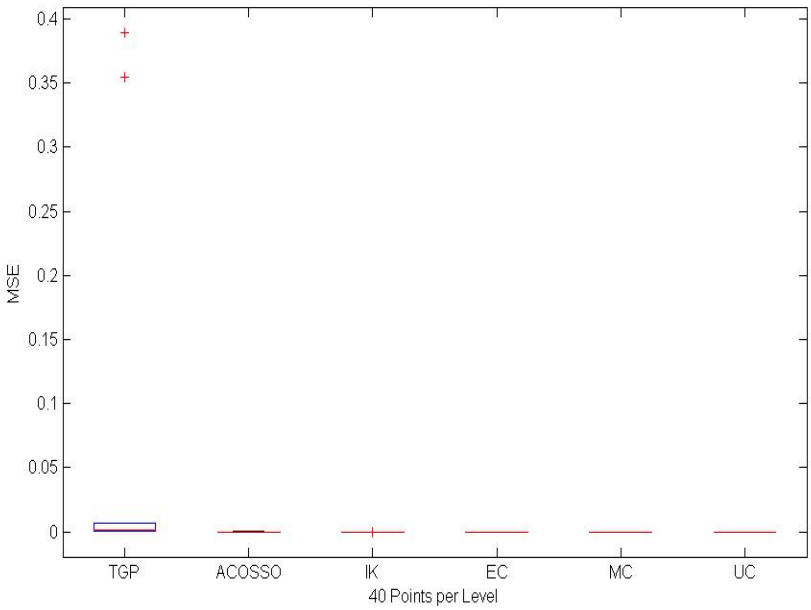

Figure 4.9. Goldstein-Price. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 40$ using the sliced LHD scheme.

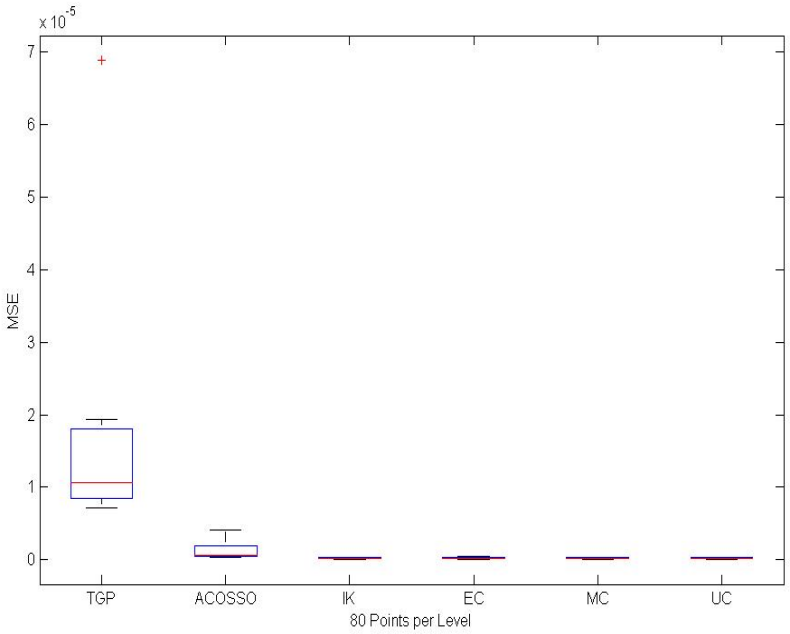

Figure 4.10. Goldstein-Price. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 80$ using the sliced LHD scheme.

4.3 Fourth Order Polynomial

The fourth order polynomial is defined in Equation 3.1.1. As mentioned, this function has 19 terms, some up to fourth order, and significant interaction terms. There are four variables; the range of the two quantitative factors is $[0, 100]$ and the two qualitative factors x_3 and x_4 have three levels, $20, 50, 80$ each.

The results for the fourth order polynomial are shown in Figures 4.11-4.14. These figures show that the Gaussian processes with the various correlation functions such as *EC*, *MC*, etc. perform well. Interestingly, ACOSSO does not seem to improve, even with the addition of points. That is, the average MSE for ACOSSO with $n = 10$ is 1.5, while the average MSE for ACOSSO with $n = 80$ is 1.4. In contrast, the other approaches all improve the MSE by eight orders of magnitude. We speculate that this is due to ACOSSO struggling when there is significant interaction between variables: it is trying to construct its response as the aggregation of separable functions which may not capture the interactions well.

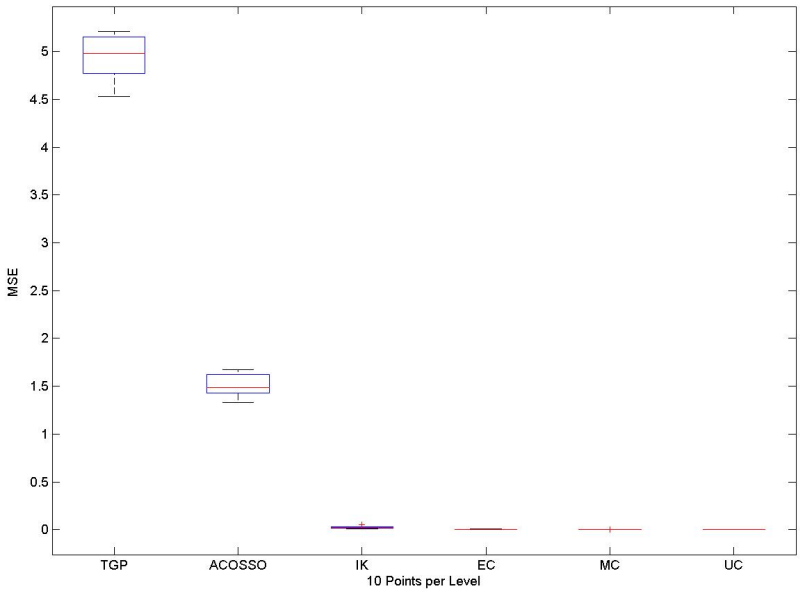

Figure 4.11. Fourth-order Polynomial. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 10$ using the sliced LHD scheme.

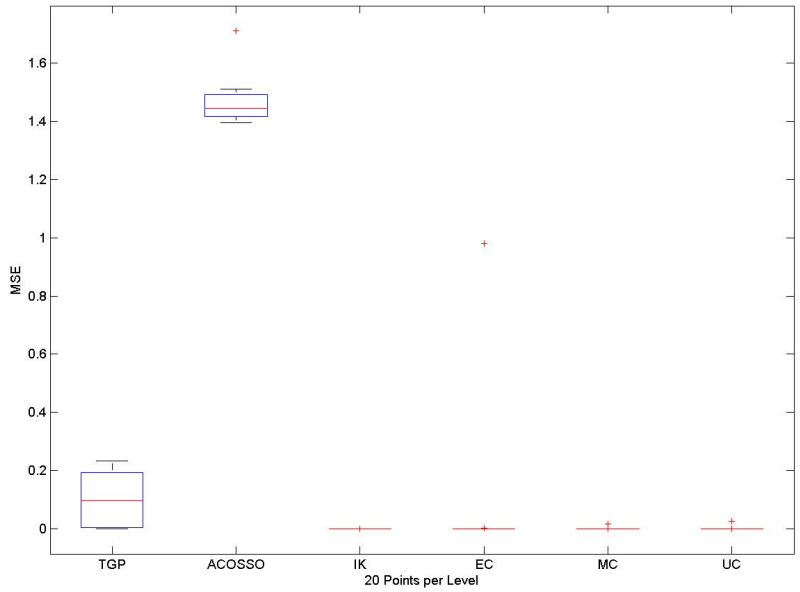

Figure 4.12. Fourth-order Polynomial. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 20$ using the sliced LHD scheme.

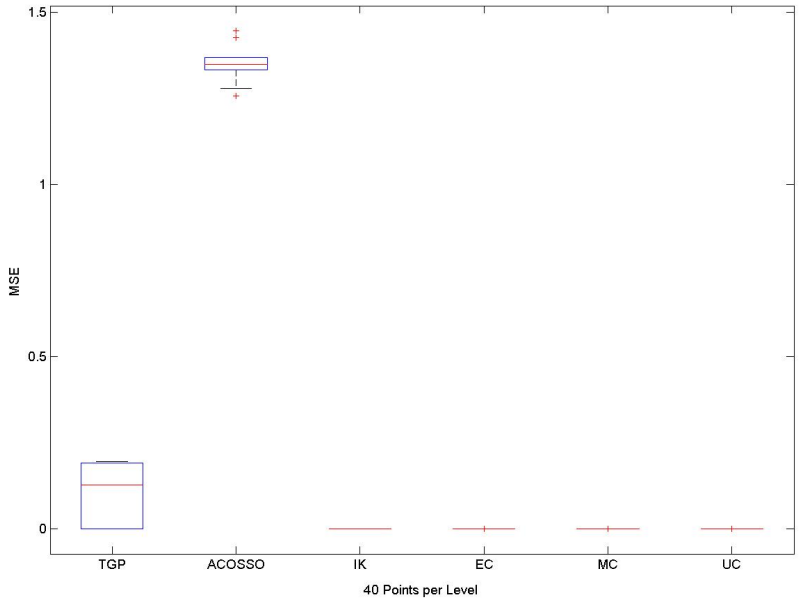

Figure 4.13. Fourth-order Polynomial. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 40$ using the sliced LHD scheme.

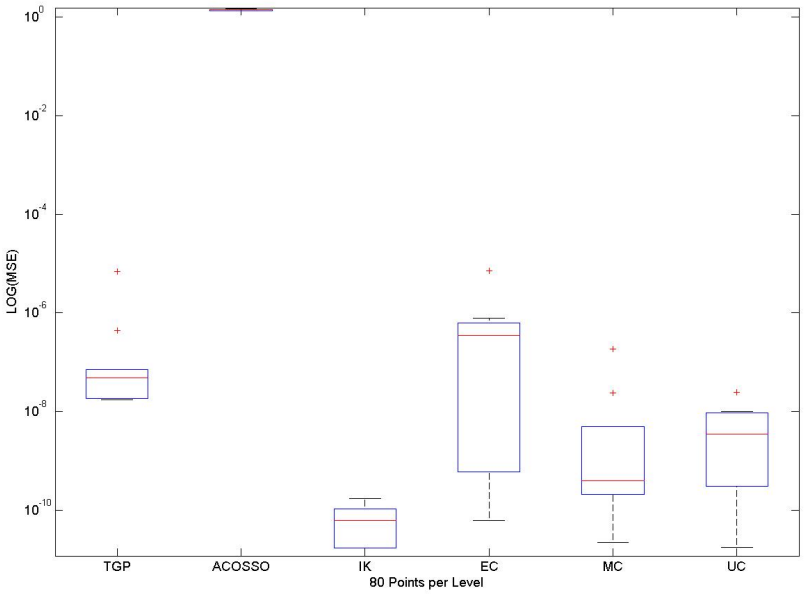

Figure 4.14. Fourth-order Polynomial. Boxplots of the MSEs for the TGP, ACOSSO, IK, EC, MC and UC methods with $n = 80$ using the sliced LHD scheme.

Chapter 5

Summary

This report investigated four main classes of surrogate methods which can handle "mixed" discrete and continuous variables: adaptive smoothing splines, Gaussian processes with special correlation functions, and Treed Gaussian processes . We were careful to use test problems which were challenging but tractable for repeated comparison runs. We did extensive comparisons, varying the number of build points used in the surrogate construction, varying the sample designs used, and building multiple surrogates of a given type so that we could compute statistics of the response to give fair comparisons (e.g. so we would not be misled by constructing only one surrogate on one set of build points).

Overall, all methods appear viable for small numbers of categorical variables with a few levels. ACOSSO and the Gaussian processes with special correlation functions generally performed well. There were subtle differences between ACOSSO and the GPs, but these two approaches both performed better than TGP, at least for Test Function 2 and the Goldstein-Price function. ACOSSO performed poorly for the fourth-order polynomial with significant interactions, but ACOSSO performed best for separable functions especially at small sample sizes. The GP with special correlation functions appears the most consistent of all the methods. However, the GP with special correlations was the most sensitive to build design and did not perform as well with a plain LHD design: the special correlation GP works best with kLHD or sliced LHD. TGP success depends on being able to identify splits where individual GPs work well in separate parts of the domain. TGP performs well on poorly scaled functions, but we found it does not perform well when the continuous variables are not predictive for certain combinations of categorical variable levels. For the fourth-order polynomial function that involves significant interaction terms, TGP performed better than ACOSSO at higher number of build points (40 or 80 build points). This is to be expected because ACOSSO is constructed over separable functions and its performance may degrade somewhat when significant interactions between variables are present.

References

[1] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Eldred, D. M. Gay, P. D. Hough, and L. P. Swiler. DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.1 user's manual. Technical Report SAND2010-2183, Sandia National Laboratories, Albuquerque, NM, 2010. Available online from `http://dakota.sandia.gov/documentation.html`.

[2] A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, Norwell, MA, 2004.

[3] P. Craven and G. Wahba. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerical Mathematics*, 31:377–403, 1979.

[4] R.L. Eubank. *Nonparametric Regression and Spline Smoothing*. CRC Press, London, 1999.

[5] R. B. Gramacy and H.K.H Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103:1119–1130, 2008.

[6] R. B. Gramacy and H.K.H Lee. Gaussian processes and limiting linear models. *Computational Statistics and Data Analysis*, 53:123–136, 2008.

[7] R. B. Gramacy and M. Taddy. Categorical inputs, sensitivity analysis, optimization and importance tempering with tgp version 2, an r package for treed gaussian process models. Technical report, R manual, `http://cran.r-project.org`, 2009.

[8] C. Gu. *Smoothing Spline ANOVA Models*. Springer-Verlag, New York, 2002.

[9] T. Hastie and R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC, London, 1990.

[10] M.C. Kennedy and A. O'Hagan. Bayesian calibration of computer models (with discussion). *Journal of the Royal Statistical Society B*, 63:425–464, 2001.

[11] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 2(12):1137–1143, 1995.

[12] Y. Lin and H. Zhang. Component selection and smoothing in smoothing spline analysis of variance models. *Annals of Statistics*, 34(5):2272–2297, 2006.

[13] S.N. Lophaven, H.B. Neilson, and J. Sondergaard. Dace - a matlab kriging toolbox. Technical report, http://www2.imm.dtu.dk/~hbn/dace/, 2009.

[14] W.R. McDaniel and B.E. Ankenman. A response surface test bed. *Quality and Reliability Engineering International*, 16:363–372, 2000.

[15] J. Neter, W. Wasserman, and M.H. Kutner. *Applied Linear Statistical Models*. Irwin, 2 edition, 1985.

[16] P.Z.G. Qian. Sliced latin hypercube designs. *submitted*, 2011.

[17] P.Z.G. Qian, H. Wu, and C.F.J. Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008.

[18] R. Rebonato and P. Jackel. The most general methodology for creating a valid correlation matrix for risk management and option pricing purposes. *The Journal of Risk*, 2:17–27, 1999.

[19] B.J. Reich, C.B. Storlie, and H.D. Bondell. Variable selection in bayesian smoothing spline anova models: Application to deterministic computer codes. *Technometrics*, 51(2):110–120, 2009.

[20] J. Sacks, W.J. Welch, T.J. Mitchel, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[21] T. Santner, B. Williams, and W. Notz. *The Design and Analysis of Computer Experiments*. New York, NY: Springer, 2003.

[22] M. Schimek, editor. *Smoothing and Regression: Approaches, Computation, and Application*. John Wiley, New York, 2000.

[23] T.W. Simpson, V. Toropov, V. Balabanov, and F.A.C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference)*, Victoria, British Columbia, Canada, September 2008. AIAA Paper 2008-5802.

[24] C.B. Storlie, H.D. Bondell, and B.J. Reich. A locally adaptive penalty for estimation of functions with varying roughness. *Journal of Computational and Graphical Statistics*, 19(3):569–589, 2010.

[25] C.B. Storlie, H.D. Bondell, B.J. Reich, and H.H. Zhang. Surface estimation, variable selection, and the nonparametric oracle property. *Statistica Sinica*, 21(2):679–705, 2010.

[26] C.B. Storlie and J.C. Helton. Multiple predictor smoothing methods for sensitivity analysis: Description of techniques. *Reliabililty Engineering and System Safety*, 93(1):28–54, 2008.

[27] C.B. Storlie, L.P. Swiler, J.C. Helton, and C.J. Sallaberry. Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models. *Reliability Engineering and System Safety*, 94(11):1735–1763, 2009.

[28] L. P. Swiler and G. D. Wyss. A user's guide to Sandia's latin hypercube sampling software: LHS UNIX library and standalone version. Technical Report SAND04-2439, Sandia National Laboratories, Albuquerque, NM, July 2004.

[29] R.J. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58:267–288, 1996.

[30] F.A.C. Viana, R.T. Haftka, and V. Steffan Jr. Multiple surrogates: How cross-validation errors can help us obtain the best predictor. *Structural and Multidisciplinary Optimization*, 39(4):439–457, 2009.

[31] F.A.C. Viana, R.T. Haftka, V. Steffan Jr., S. Butkewitsch, and M.F. Leal. Ensemble of surrogates: a framework based on minimization of the mean integrated square error. In *Proceedings of the 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages AIAA Paper 2008–1885, Schaumburg, IL, April 2008.

[32] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics, 1990.

[33] Z. Wu. Multivariate compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4:283–292, 1995.

[34] Q. Zhou, P.Z.G. Qian, and S. Zhou. A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics*, 53:266–273, 2011.

Appendix A

Additional Test Framework Functions

The first of the two remaining functions in the testbed has one categorical variable with five levels. It also has one continuous variable which falls between the values of 0 and 1. This function has a region where the responses at the different categorical levels are very similar. This will allow us to evaluate how well the different surrogate approaches can resolve the different levels.

$$f(x) = \begin{cases} 3.5(x_2+0.5)^4 & \text{if } x_1 = 1 \\ 0.5(x_2+0.5)^4 & \text{if } x_1 = 2 \\ 2.5(x_2+0.5)^4 & \text{if } x_1 = 3 \\ 0.7(x_2+0.5)^4 & \text{if } x_1 = 4 \\ (x_2+0.5)^4 & \text{if } x_1 = 5 \end{cases}$$

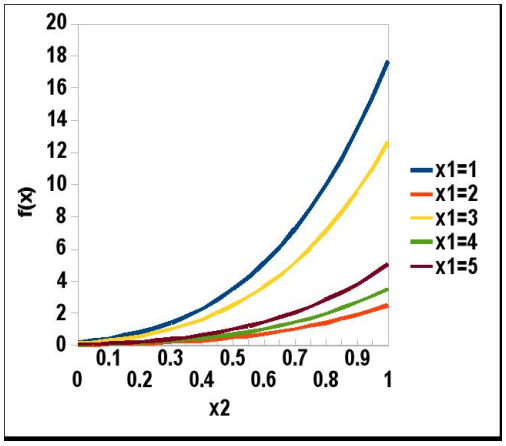

Figure A.1. Test Function 1

The last function has an arbitrary number of variables. Furthermore, the number of continuous variables relative to the number categorical variables is arbitrary as is the number of levels for each categorical variables. This will allow us to test the scalability of the surrogate approaches with respect to both the number of variables and the number of levels per categorical variable.

$$f(x) = \sum_{i=1}^{n} (x_i - 1)^4$$

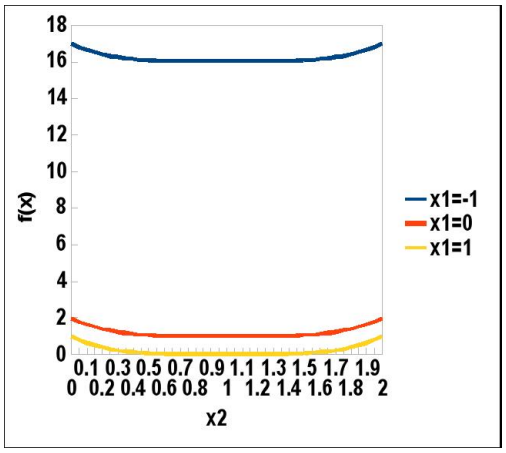

Figure A.2. Test Function 3

Appendix B

Results of Scaling Studies

In addition to the evaluations described in the main body of this paper, we did some preliminary studies to assess the potential for scalability of these mixed-variable surrogate modeling approaches with respect to the number of discrete/categorical variables and to the number of levels per variable. These studies confirmed that the curse of dimensionality affects all of these approaches and that there is no clear path to scalability. However, we include the results in this appendix in the interest of completeness.

As a baseline, we include results for categorical regression[15]. Until recently, this was the only option in the literature for constructing response surface surrogates with mixed variables. Since we did not describe this method in the main body of the paper, we present it briefly here. In short, this approach constructs a separate response surface over the continuous parameters for each categorical level. More formally, it is described using indicator functions as in the following simple example:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2,$$

where X_1 is continuous, X_2 is binary and

$$Y = \beta_0 + \beta_1 X_1 \text{ for } X_2 = 0 \text{ and} \tag{B.0.1}$$
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 \text{ for } X_2 = 1 \tag{B.0.2}$$

While intuitive and straightforward to implement, categorical regression is computationally expensive. It is necessary to collect data at enough samples over the continuous variables for *each* discrete combination for an accurate regression function. Therefore, as the number of discrete variables and/or the number of categorical levels per discrete variable increases, there is a combinatorial explosion in the number of simulations that must be run. We use categorical regression as a baseline for benchmarking the other three mixed variable surrogate approaches, but we consider it feasible to use in practice only for very small problems.

For our scalability studies, we used the following function from our testbed:

$$f(x) = \sum_{i=1}^{n} (x_i - 1)^4.$$

Our initial test had four variables, two of which were continuous with values between 0 and 2 and two if which were categorical with three levels (0, 1, and 2). The mean squared errors for

the different surrogate approaches with different numbers of build points are shown in Figure B.1. From this we can see that even with only four variables, the performance of categorical regression

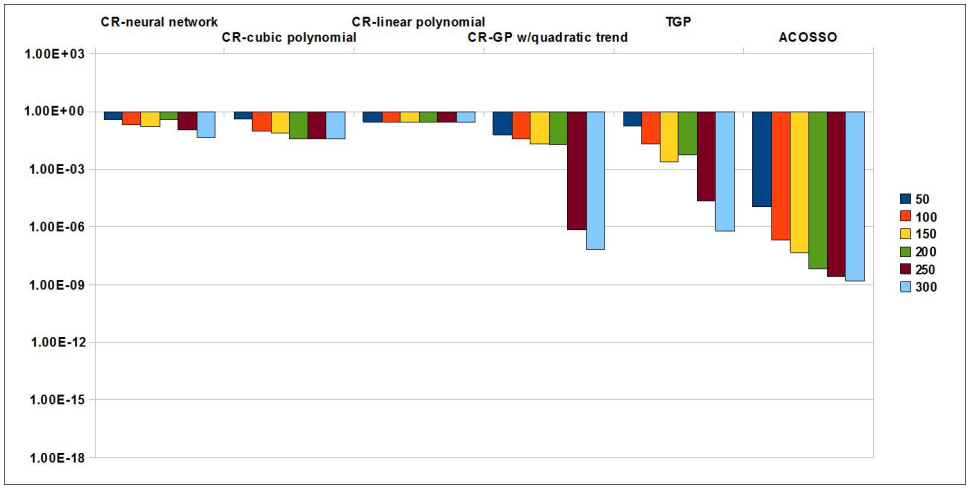

Figure B.1. This figure shows mean squared error values for a variety of mixed variable surrogate approaches with different numbers of build points. The problem has two continuous variables and two categorical variables. The poor performance of categorical regression demonstrates that scalability is extremely limited even for small problems.

.

is generally poor and does not improve significantly with increasing number of build points. TGP and ACOSSO performed reasonably well, so we completed some follow-on studies using just these two approaches.

We next considered scalability with respect to the number of levels per categorical variable. Tables B.1 and B.2 show the results for increasing the number of categorical levels from three to five. In both cases, the methods scale better with respect to the number of categorical variables than with respect to the number of categorical levels. This is reflected in larger increases in mean squared error as the number of categorical levels increases versus as the number of categorical variables increases. Additionally, increases the number of build points improves the mean squared error at a noticeably faster rate for larger numbers of categorical variables than for larger numbers of categorical levels.

Table B.1. This table shows the mean squared error for TGP given different numbers of categorical variables, different numbers of categorical levels per variable, and different numbers of build points. Scalability deteriorates faster as the number of categorical levels increases.

number of build points	2 categorical, 3 levels	2 categorical, 5 levels	5 categorical, 3 levels	5 categorical, 5 levels
50	0.7217199	119.75	1.35	319.22
100	0.03391995	57.15	0.79	300.08
150	0.01617074	25.94	0.87	272.76
200	0.00631333	25.26	0.72	265.96
300	$4.45 * 10^{-5}$	17.91	0.52	231.41
500	$1.74 * 10^{-6}$	1.27	0.32	223.68

Table B.2. This table shows the mean squared error for ACOSSO given different numbers of categorical variables, different numbers of categorical levels per variable, and different numbers of build points. Scalability deteriorates faster as the number of categorical levels increases.

number of build points	2 categorical, 3 levels	2 categorical, 5 levels	5 categorical, 3 levels	5 categorical, 5 levels
50	$1.20 * 10^{-4}$	$8.15 * 10^{-4}$	$2.24 * 10^{-4}$	$2.56 * 10^{-1}$
100	$9.31 * 10^{-6}$	$1.55 * 10^{-3}$	$6.27 * 10^{-6}$	$4.06 * 10^{-6}$
150	$1.50 * 10^{-6}$	$1.34 * 10^{-3}$	$2.01 * 10^{-6}$	$2.56 * 10^{-4}$
200	$1.75 * 10^{-7}$	$3.20 * 10^{-6}$	$6.97 * 10^{-7}$	$1.99 * 10^{-3}$
300	$3.17 * 10^{-7}$	$4.68 * 10^{-5}$	$1.31 * 10^{-7}$	$5.24 * 10^{-5}$
500	$7.69 * 10^{-8}$	$3.08 * 10^{-4}$	$8.56 * 10^{-8}$	$2.14 * 10^{-5}$

DISTRIBUTION:

1 Herbert Lee
Applied Mathematics and Statistics
University of California, Santa Cruz
Basked School of Engineering
1156 High St, MS SOE2
Santa Cruz, CA 95064

1 Peter Qian
Department of Statistics
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706

1 Curtis Storlie
Statistics Department
Los Alamos National Laboratory
P.O. Box 1663, MS F600
Los Alamos, NM 87545

1 Xu Xu
Department of Statistics
University of Wisconsin-Madison
1300 University Ave.
Madison, WI 53706

1 MS 0829 B.M. Rutherford, 00431
1 MS 0748 J.C. Helton, 01514
1 MS 1327 W.E. Hart, 01464
1 MS 1318 J.R. Stewart, 01441
1 MS 1318 B.M. Adams, 01441
1 MS 0670 K. Dalbey, 05562
1 MS 1318 M.S. Eldred, 01441
1 MS 1318 L.P. Swiler, 01441
1 MS 1318 T.G. Trucano, 01440
1 MS 1326 C.A. Phillips, 01465
1 MS 1326 M.D. Rintoul, 01465
1 MS 1326 J.D. Siirola, 01465
1 MS 1326 J.P. Watson, 01465
1 MS 0812 V.J. Romero, 01544
1 MS 1233 A.A. Giunta, 05952
1 MS 9159 L.E. Bauman, 08954
1 MS 9159 K.T. Carlberg, 08954
1 MS 9159 G.A. Gray, 08954
1 MS 9159 P.D. Hough, 08954
1 MS 9159 J. McNeish, 08954

1	MS	9155	J. Ruthruff, 08954
1	MS	9159	C. Safta, 08954
1	MS	9409	J.A. Templeton, 08365
1	MS	0899	RIM - Reports Management, 9532 (electronic copy)

Sandia National Laboratories